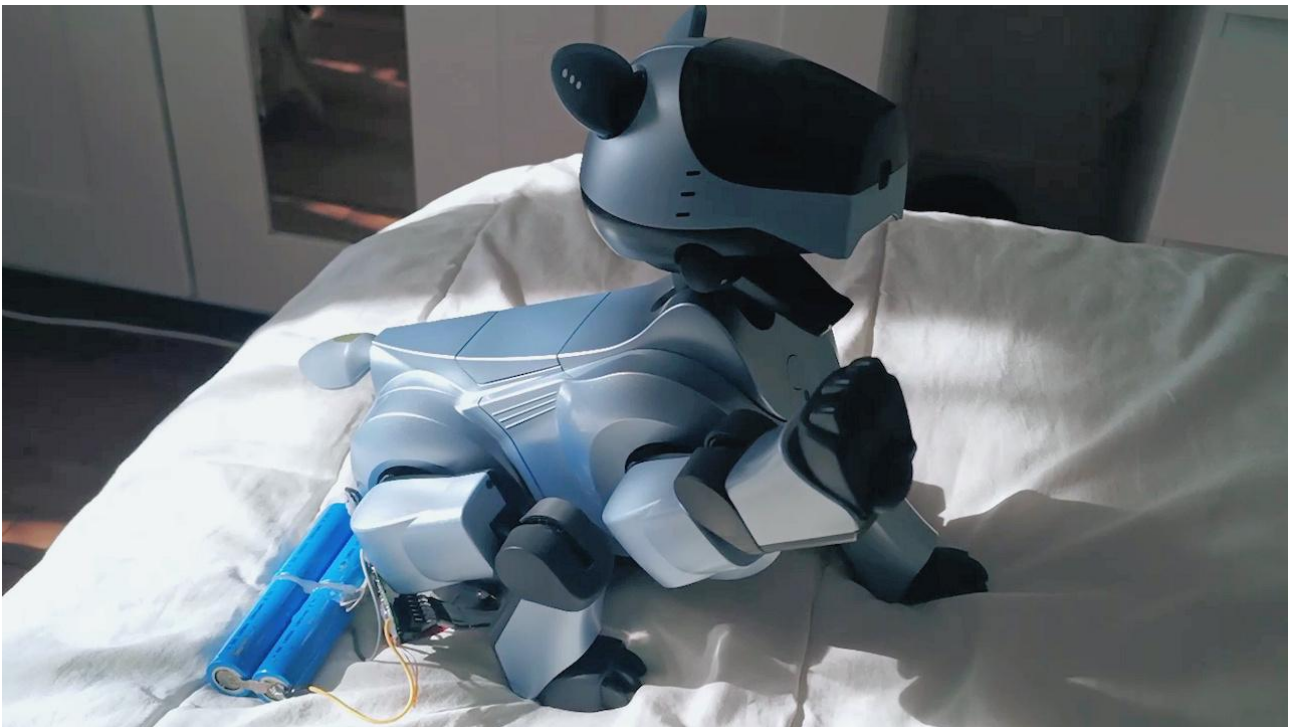# ERA-201B1 re-cell tutorial

How to repair a Sony Aibo ERS-2xx robot battery pack

Abstract

This tutorial is made for happy owners of Sony Aibo robots, more particularly models from the years 2000-2003 including ERS-210/ERS-210A and ERS-220/ERS-220A, who would like to repair a used ERA-201B1 battery pack themselves.

Disclaimer

This document and its contents are intended to people enough qualified and experienced in electronics. The information it contains is the result of trial and error which has led to an efficient method of renewing some of the Aibo robot's batteries. This is one way to achieve it and there are certainly others that can work as well.

The procedure is quite technical even for initiated. If you know that you are not yet experienced or you do not have the necessary equipment, it is not recommended that you undertake this operation yourself, as you may permanently damage your battery or even suffer a fire in case of Li-ion cell short circuit.

BattMon 2.0 repository, which includes this tutorial, is provided on Github under MIT license. Its author cannot be held responsible for the consequences that you will make of its use. Be careful, read this document in its entirety and take precautions before you start.

Version history

Revision	Date	Changes	Author
1.0	2022-02-27	Initial release	Loïc POLLIER
1.1	2022-03-27	Fix typo in EEPROM I2C test point reference and add comment for other Sony Aibo batteries with bq2060	Loïc POLLIER
1.2	2022-04-27	Fix typo in section Reset the EEPROM parameters	Loïc POLLIER
1.3	2022-09-27	Add IC21 component identification discovery	Loïc POLLIER

Table of contents

1. Gather tools and equipment

In addition to a solid background in electronics, you need tools and equipment to perform the complete battery replacement operation. Below is the minimum set.

Minimum set:
- Electronic tools (flat screwdriver, flat pliers, cutting pliers, multimeter, electric wires, insulating tape, soldering iron, tin coil, spot welding machine, nickel roller).
- 1 I2C EEPROM memory programmer (e.g. Bus Pirate) connected to a computer.
- 4 new 18500 Lithium-ion cells.

Moreover, this recommended set, in addition to the minimum set, will certainly guarantee a higher success rate by measuring the actual capacity of the battery and checking its characteristics before, during and after the repair.

Recommended set:
- 1 SMBus battery monitoring toolkit (e.g. BattMon 2.0).
- 1 Li-ion battery charger (e.g. HTRC B6).
- 1 SO-8 IC test clamp.
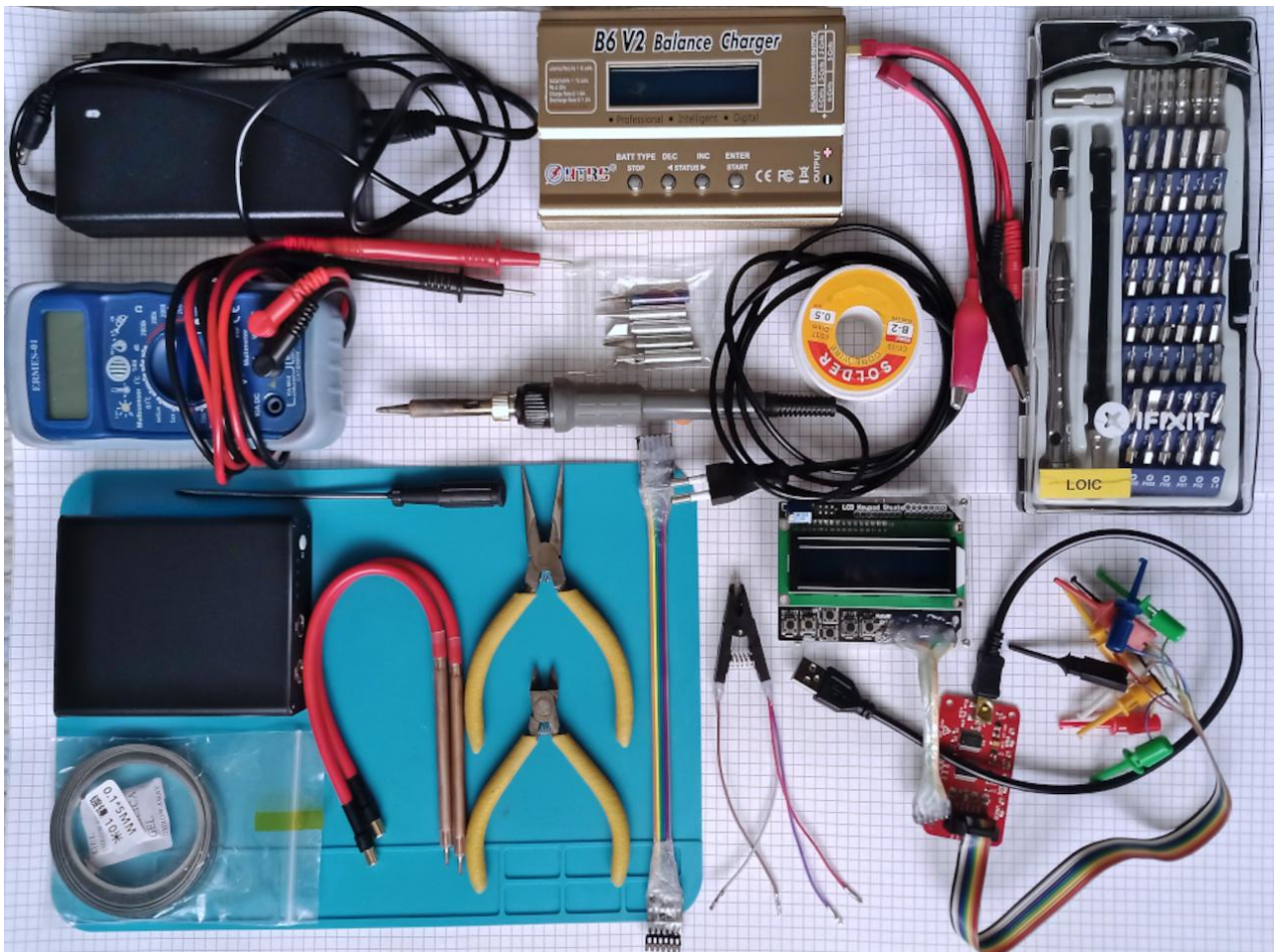
You can find links in Appendices A and B on how to obtain the equipment references, the datasheet for the main components and the tools documentation.

To summarize, the repair can be broken down into three stages: Replace the cells; Reset some of the EEPROM parameters; Reinitialize the first level battery protection.

2. Open the battery case and remove the old cells

Now that you are equipped, we can start by opening the case to extract the battery pack and remove the used cells. It is recommended that the pack is completely discharged.

This step can be done with a fairly wide and thin flat screwdriver. But to avoid damaging the case too much, you can also use a thin and long flat blade by gently levering it along its entire length.
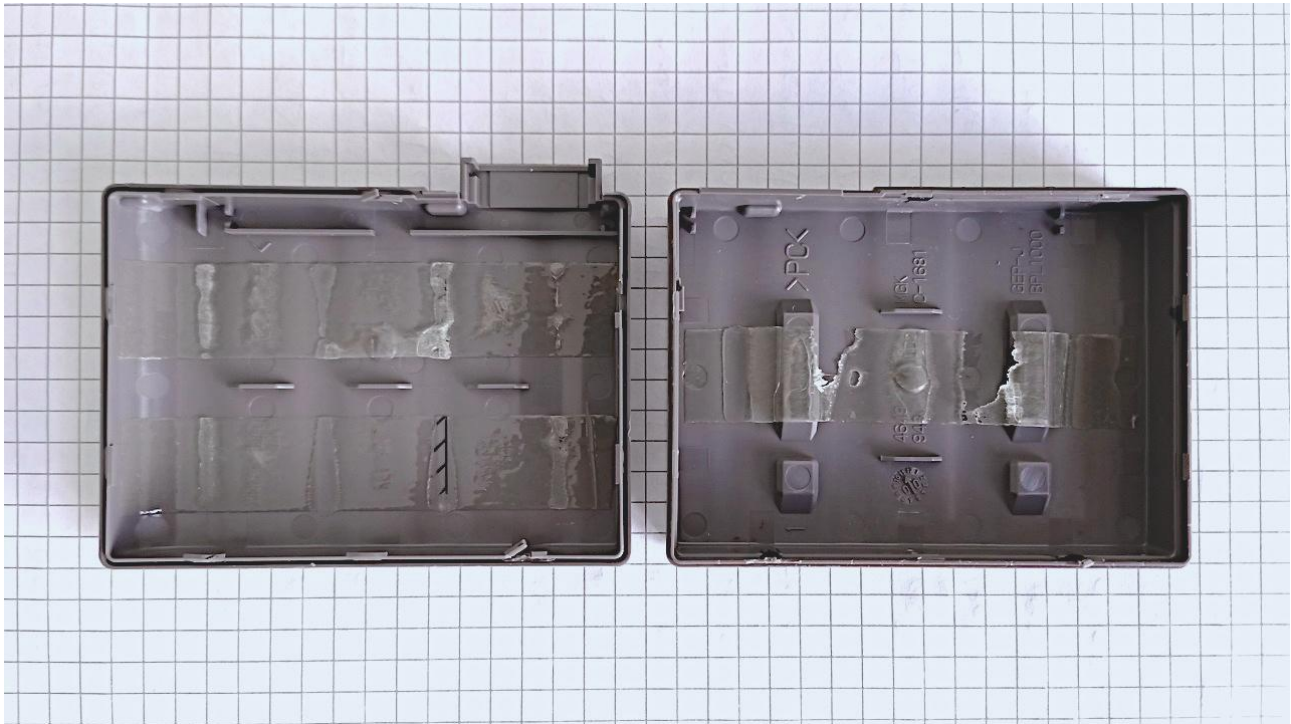

What holds the two parts of the case around the cells are both strong tape and nine retaining clips on all sides. It is pretty hard not to break a few of them. Take your time.

3. Weld the new cells and reassemble them in pack

After taking the pack out of its case, you can unsolder the nickel plate (BAT+) and the two gray (BATM) and black (BAT-) wires from the PCB. Watch out for short circuits here.
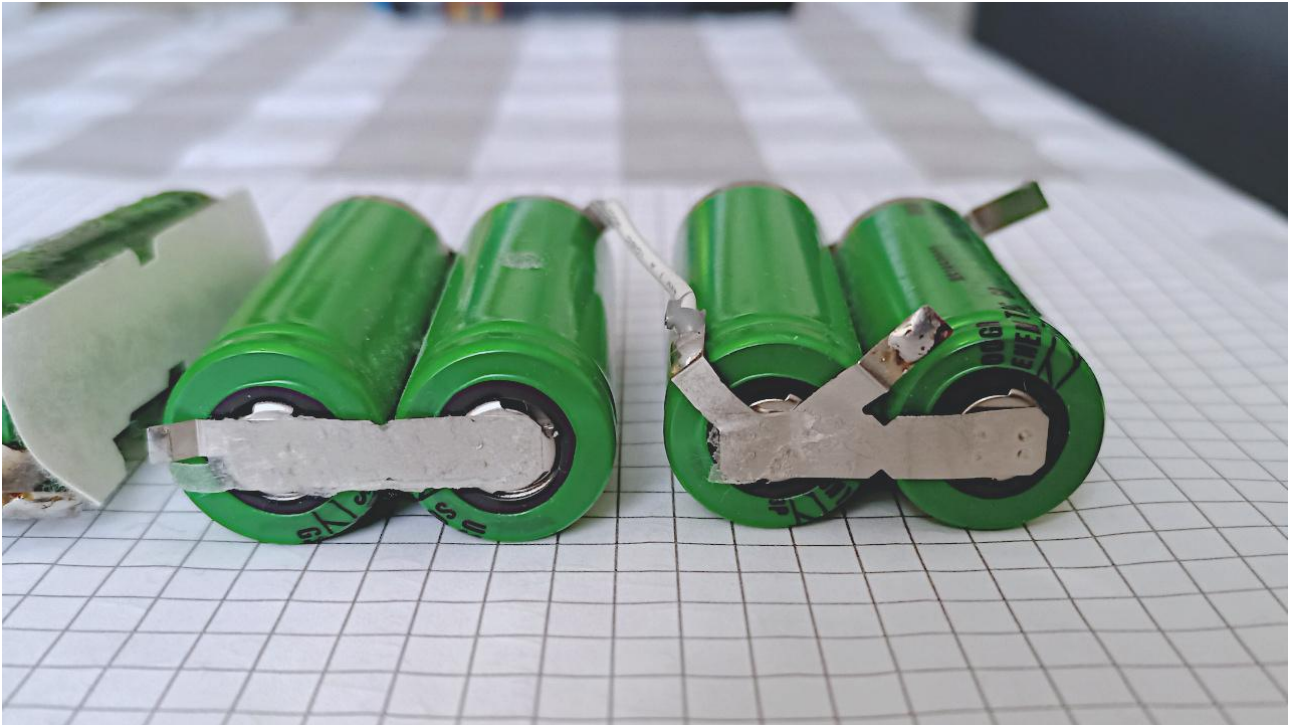

If you have a Lithium-ion battery charger, it is time to individually measure the real capacity of each new cell and thus verify that it is at least as good as the original 1150mAh.

Depending on the battery manufacturer, it often happens that the actual capacity of cheap cells is much lower than that indicated. It is advisable to use brand cells like NCR18500A which have a real capacity close to 2000mAh, as well as a lower internal resistance.

Once the old cells are separated from the PCB, you can continue by reassembling the pack with the new ones. It is better to solder the cells in the same way as the original. For this, you can use a spot welder instead of a soldering iron, to reduce thickness and heat damage. It is essential to add a layer of electrical insulation such as Kapton tape. You can also reuse the original insulators made of thicker adhesive paper to better wedge the cells.
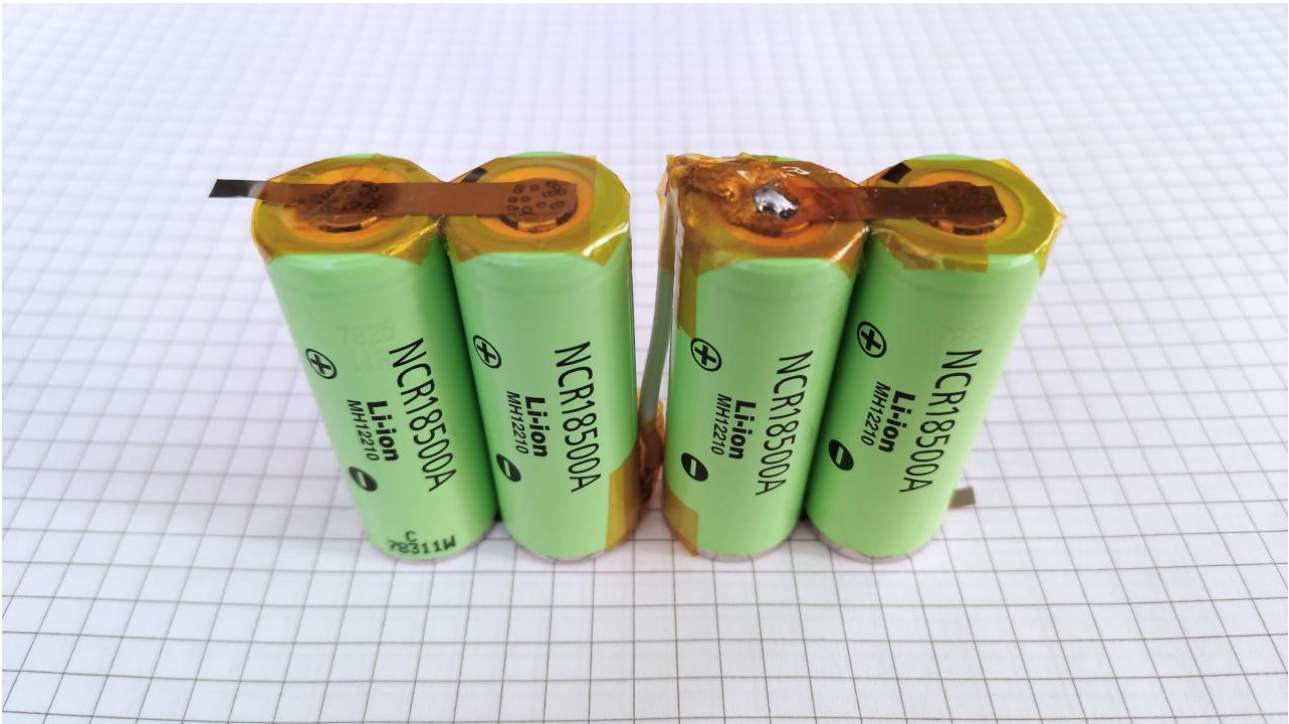
Finally, solder the white wire to serially connect the two sets of cells welded in parallel.

4. Reset the EEPROM parameters

We are now going to update some of the parameters saved in the 24LC01 EEPROM memory IC42 which are used by the bq2040 gas gauge IC41.
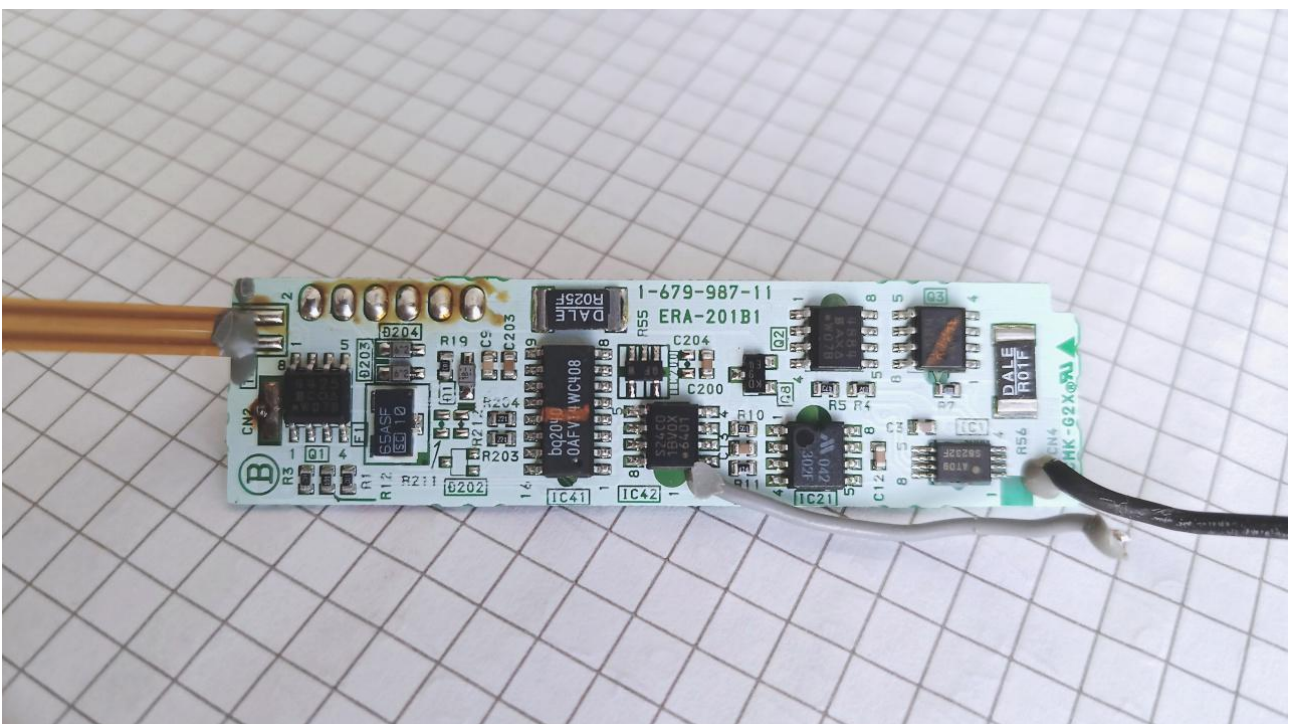
To do this, you will have to use a memory programmer such as the Bus Pirate, interfaced to the EEPROM IC42 via the I2C bus.

As indicated in the 24LC01 datasheet, four wires must be connected to achieve the physical interface: VCC (+3.3V), VSS (0V), SCL (I2C Clock) and SDA (I2C Data/ Address). You can either solder the necessary wires to test points TP202, TP209, TP203 and TP204 respectively, or you can create a test clamp in SO-8 format, like the one below.
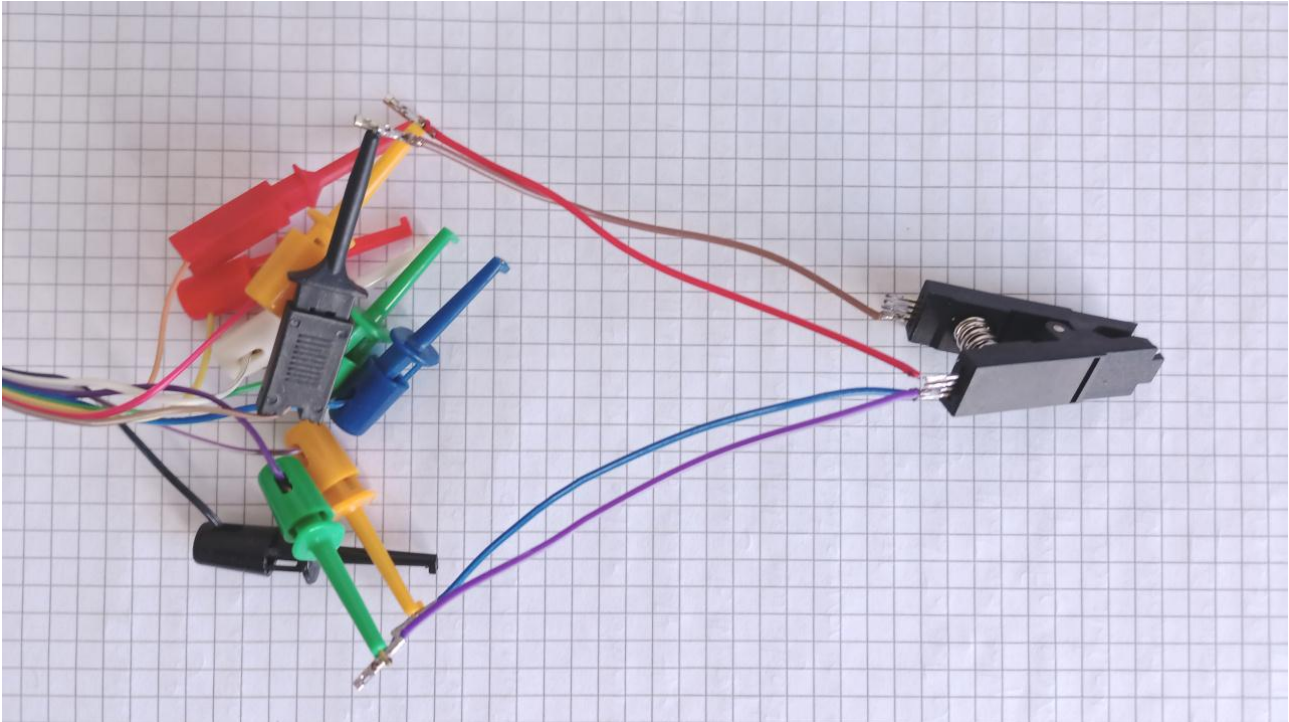

In this case, connect it well so that the pins coincide correctly. If necessary, you can refer to 24LC01 datasheet, where the direction and polarity of the EEPROM is indicated relative to the position of pin 1 represented by a dot or notch.
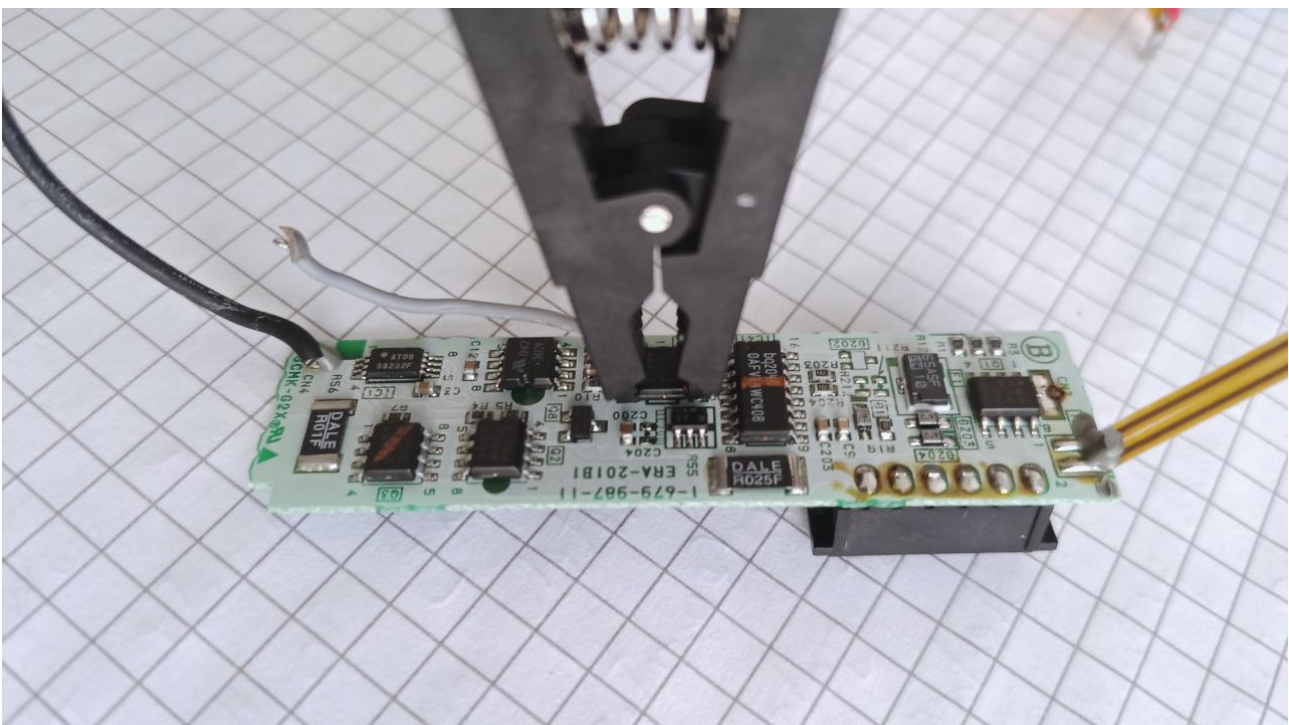
Afterwards, you can plug the Bus Pirate USB cable to your computer. Here we will use a Windows PC with PuTTY software installed, but it is pretty much the same directly with a Mac or Linux terminal.
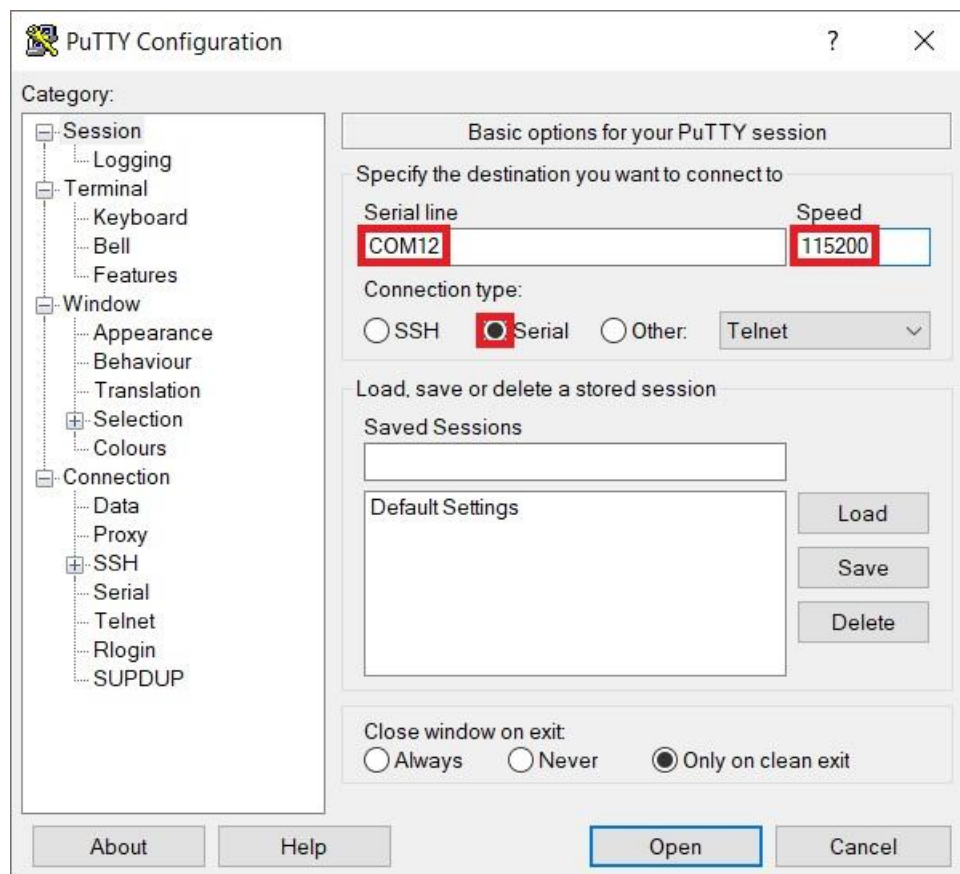
This involves connecting to the Bus Pirate by selecting the "Serial" option with the default values of 115200 bit/s in "Speed" and your USB COM port number in "Serial line". To find out your port number, go to the Windows menu "Device Manager" > "Ports (COM & LPT)", which is in this example COM12.
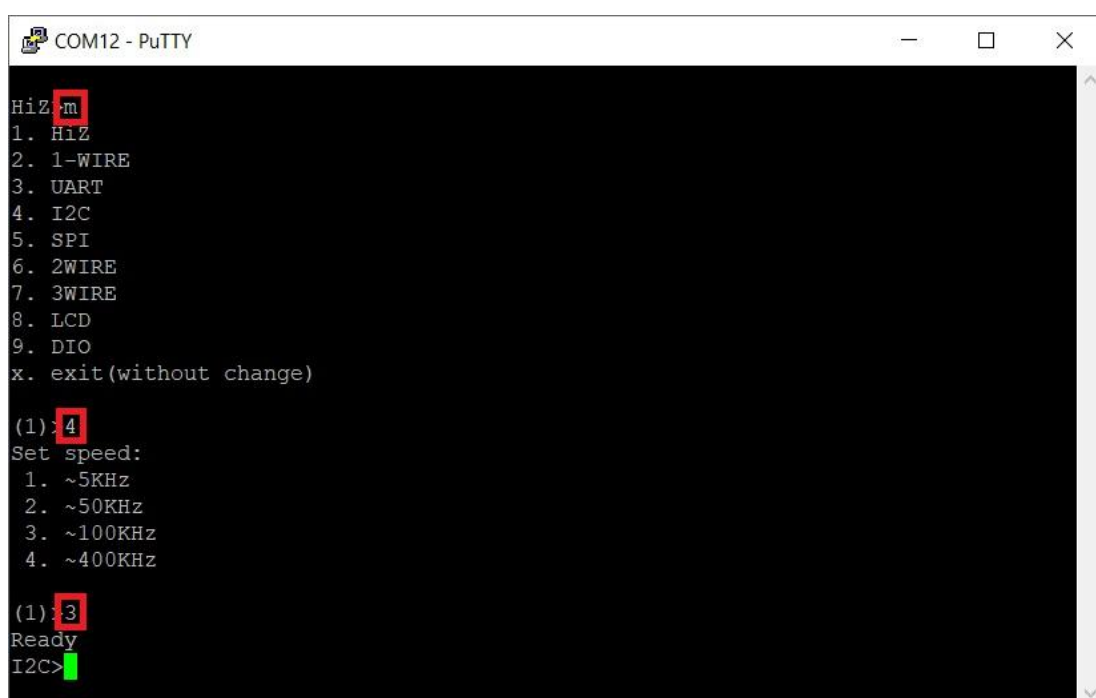
We will now communicate with the Bus Pirate by command line. Click the "Open" button and start by entering the "m" command to display the main menu as above. You must then select I2C by entering "4" and a standard speed rate of 100kHz by entering "3".

If all went well, you can supply IC42 with the "W" command (or cut off the power with "w"). And "v" allows you to check the voltage which should be close to 3.3V in "2.(RD)" column.

Enter the command "(1)" to verify that the EEPROM memory is present on the I2C bus. You should see a response like this: "0xA0(0x50 W) 0xA1(0x50 R)...".

You can read the memory contents with this command: "[0xA0 0x00 [0xA1 r:0x65]". Be sure to copy-paste this original data into a separate text file, you may find it useful later.

Among the contents of the memory you can see surrounded in orange three parameters that we are going to modify: Cycle Count, Design Capacity and Full Charge Capacity.

So far we have sent read commands to memory, we will now have to write in it. The values are coded in hexadecimal, so we have to convert from decimal to hexadecimal the value we want to write. Most calculators, like the one in Windows, offer this function.

The initial values in this example are: Cycle Count: "0x002A" -> 42 full charge cycles, Design Capacity: "0x08FC" -> 2300mAh, Full Charge Capacity: "0x08FC" -> 2300mAh.

Imagine that we individually measured our four cells with an actual capacity of around 1500mAh each. It will therefore be necessary to write in the EEPROM 3000mAh, the double corresponding to the total capacity of the pack, so 0x0BB8 in hexadecimal.

In addition, the charge cycle counter must also be reset. For all this, we will enter this series of commands, as seen in red above:

"[0xA0 0x0E 0x00]" // Cycle Count LSB set to 0x00 at memory address 0x0E
"[0xA0 0x0F 0x00]" // Cycle Count MSB set to 0x00 at memory address 0x0F
"[0xA0 0x10 0xB8]" // Design Capacity LSB set to 0xB8 at memory address 0x10
"[0xA0 0x11 0x0B]" // Design Capacity MSB set to 0x0B at memory address 0x11
"[0xA0 0x60 0xB8]" // Full Charge Capacity LSB set to 0xB8 at memory address 0x60
"[0xA0 0x61 0x0B]" // Full Charge Capacity MSB set to 0x0B at memory address 0x61

Well adjust the capacity in memory to twice that of your cells. If you have NCR18500A cells of around 2000mAh, you will need to use 4000mAh, so 0x0FA0 in hexadecimal.

Finally, check the new memory contents: "[0xA0 0x00 [0xA1 r:0x65]". You should see the updated values as in green below.

In the end, you can turn off the power with "w" and unplug everything. There are other settings in memory, but it is safe not to change them.

But if you find that you made a mistake, you can always rewrite the contents of your EEPROM with the original data you saved in a text file. Each command starting with "0xA0" writes a value in hexadecimal to an address in hexadecimal in this way:

"[0xA0 address value]"
Here is an example to write value 0xAB in a memory at address 0x65 and then read it:

"[0xA0 0x65 0xAB]"
"[0xA0 0x65 [0xA1 r]"

According to the datasheet of the bq2060 gas gauge, it seems possible to achieve the same operation with the ERA-301B1 and ERA-7B1/ERA-7B2 batteries. In this case, the Cycle Count parameter is at the same EEPROM address 0x0E/0x0F, while Design Capacity becomes Pack Capacity at 0x3A/0x3B and Full Charge Capacity becomes Last Measured Discharge at 0x38/0x39. The following step is specific to the ERA-201B1 and therefore must be adapted in case of another battery pack.

5. Solder the pack to the board and reinitialize the battery protection

Before closing the case, the last step is to solder the pack to the PCB and reinitialize the reference of the first level protection circuit IC1 between BAT- and GND/PACK- voltages, which consists of simultaneously short-circuiting the source of transistors Q2 and Q3. <u>However, if you do it directly, you will probably destroy fuse F1 via transistor Q8.</u>

To avoid this pitfall, <u>you need to temporarily solder a wire between test points TP4/TP5 and TP12</u>. Only then you can solder the nickel plate (BAT+), the gray (BATM) and black (BAT-) wires to the PCB as initially. With your multimeter, verify the voltage at the terminals of the two sets of cells to respect the polarity. <u>Feel free to reuse insulating tape in between.</u>

Now, you should measure with your multimeter, that the resistance between TP4/TP5 (BAT-) and TP24 (GND/PACK-) is very high. If this is true, then you can use another wire to connect them for a very short time (a split second). IC1 reinitialization should be done.

Finally, measure again that the resistance between TP4/TP5 and TP24 is now very close to 0Ω, and unsolder the wire between TP4/TP5 and TP12. If you have the BattMon 2.0 toolkit, you can also check that the battery is working well and returning data such as Voltage, Current, State of Charge, Temperature, Remaining Capacity or Cycle Count.
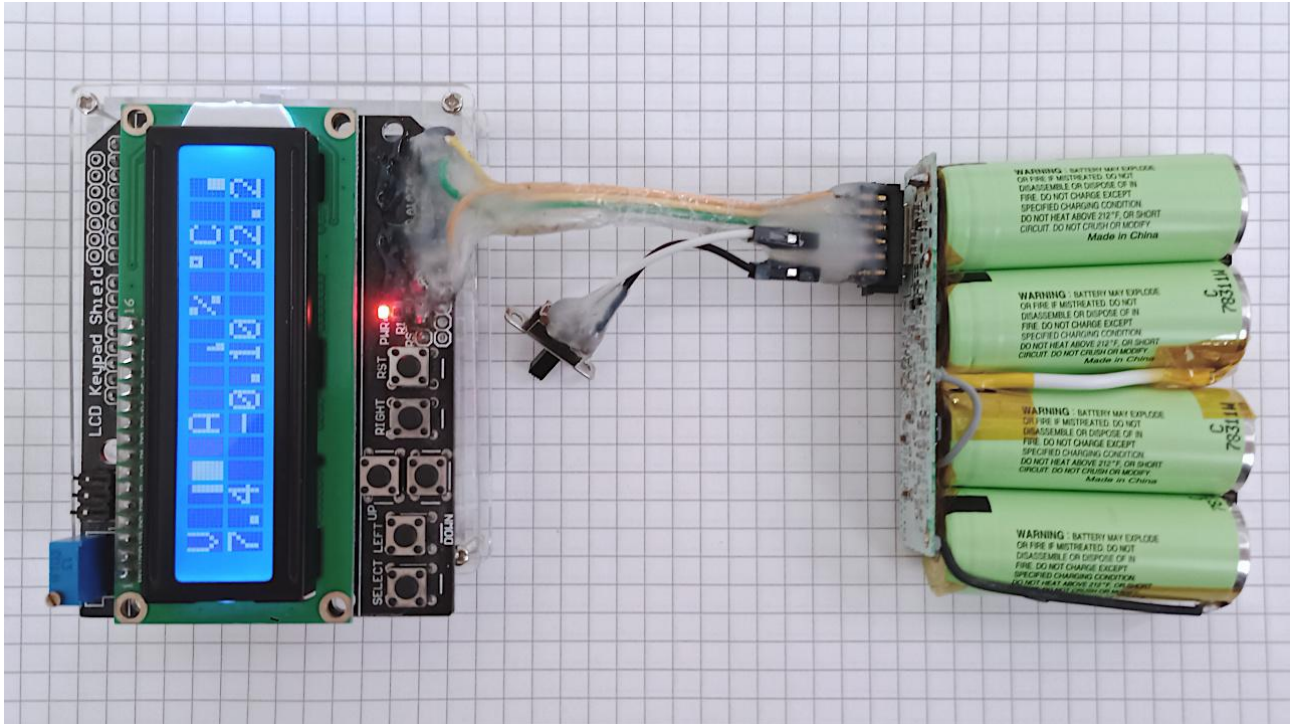

6. Reintegrate all in the case

For the final test, you have to put the pack back in its case, positioning the connector first.

Tips and tricks

To conclude, it will be useful to perform a full charge and then a full discharge, so that the bq2040 gas gauge IC41 determines the exact capacity of the battery. This calibration can be done either via the charger in discharge mode or with the BattMon 2.0 toolkit, when the voltage reaches a value between EDV1 (6.3V) and EDVF (6.0V), as indicated in the datasheet: "FCC is updated on the first charge after a qualified discharge to EDV1".
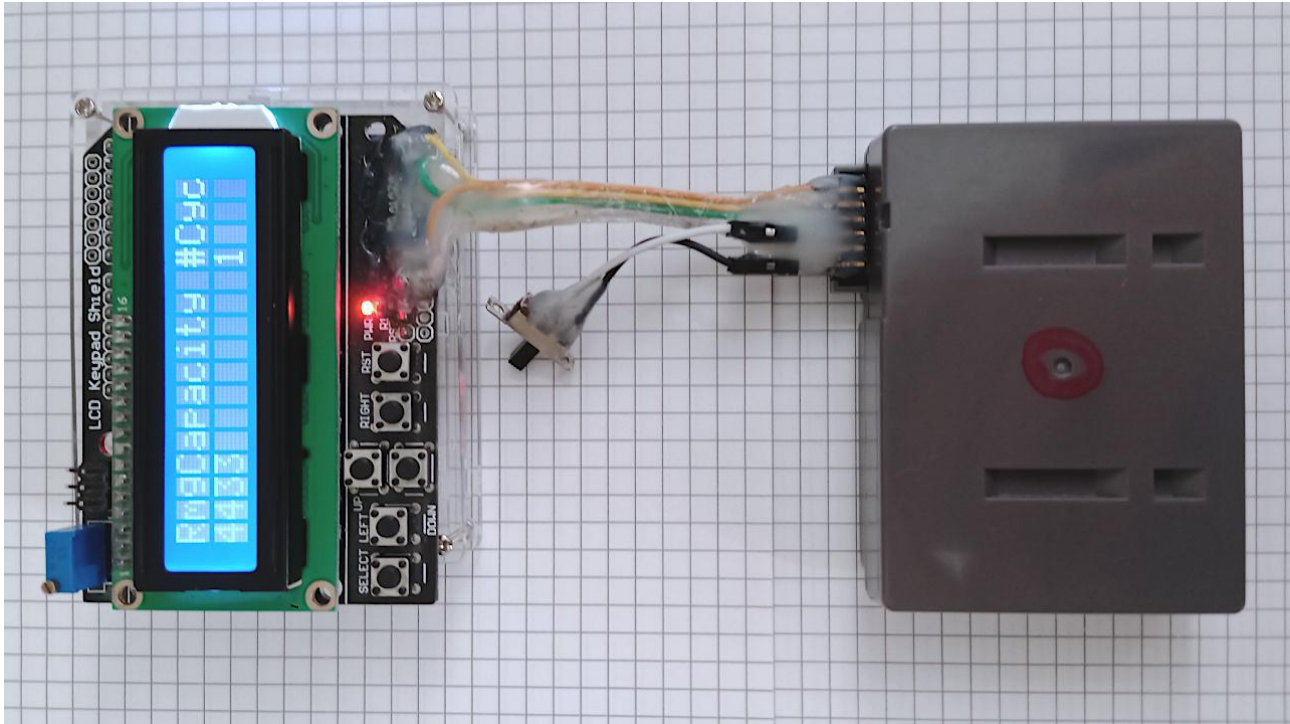
I accomplished this operation on four batteries. Two packs have been replaced with cells of around 1500mAh and two others with NCR18500A cells of around 2000mAh. I burned the fuse F1 several times, hopefully most of the components can still be found on internet.

Appendix A

All equipment is also easily found on the internet, here is the order price for some of them:

Spot welding machine + nickel roller: ~30$/€/F from Aliexpress / ~50$/€/F from Amazon
Kapton tape: ~3$/€/F from Aliexpress / ~8$/€/F from Amazon
HTRC B6 charger: ~25$/€/F from Aliexpress / ~40$/€/F from Amazon
Bus Pirate v3.6 + basic probe set: ~40$/€/F from Aliexpress or Seeed Studio
https://www.seeedstudio.com/Bus-Pirate-v3-6-universal-serial-interface-p-609.html
Arduino Uno R3: ~10$/€/F from Aliexpress / ~20$/€/F from Arduino
http://store.arduino.cc/products/arduino-uno-rev3
LCD Keypad Shield: ~5$/€/F from Aliexpress / ~10$/€/F from Arduino
https://www.dfrobot.com/product-51.html

Appendix B

Here are the main components datasheet and tools technical documents that should interest you to understand what is explained in this tutorial:

bq2040 and bq2060 gas gauge datasheet:
https://www.ti.com/lit/ds/symlink/bq2040.pdf
https://www.ti.com/lit/ds/symlink/bq2060.pdf
24LC01 EEPROM memory datasheet:
https://ww1.microchip.com/downloads/en/devicedoc/21711c.pdf
Bus Pirate specification:
http://dangerousprototypes.com/docs/Bus_Pirate
PuTTY terminal emulator:
https://www.putty.org
Arduino Uno R3 specification:
https://docs.arduino.cc/hardware/uno-rev3
Arduino Uno R3 schematic:
https://content.arduino.cc/assets/UNO-TH_Rev3e_sch.pdf
LCD Keypad Shield specification:
https://wiki.dfrobot.com/LCD_KeyPad_Shield_For_Arduino_SKU__DFR0009
LCD Keypad Shield schematic:
https://image.dfrobot.com/image/data/DFR0009/LCDKeypad%20Shield%20V1.0%20SCH.pdf

And here is the result of my research to make this battery work again:

ERA-201B1 wiring diagram with BattMon 2.0 toolkit:
https://github.com/lpollier/battmon/blob/master/example/ERA-201B1_wiring_diagram.png
ERA-201B1 reverse engineering schematic:
https://github.com/lpollier/battmon/blob/master/example/ERA-201B1_reverse_engineering_schematic.pdf
Identifying a SOIC-8 IC with "302F" marking:
https://electronics.stackexchange.com/questions/550228/identifying-a-soic-8-ic-with-302f-marking
The video of my first test made with a re-celled battery:
https://youtu.be/zO9p-0TFVrM

I want to say a special thanks to Chris, Andrej and Andrew for helping me improve this tutorial and Furrtek for identifying the second level protection circuit IC21.